Chapter 9

Peripheral Functional Units

CHAPTER 9. PERIPHERAL FUNCTIONAL UNITS

9. PERIPHERAL FUNCTIONAL UNITS

The DSP3210 provides four on-chip peripheral units; serial I/O (SIO), DMA controller (DMAC), 32-bit timer (TIMER), and bit I/O (BIO). Access to peripheral functional units is memory-mapped. Detailed descriptions of these facilities are contained in this section.

9.1 Serial I/O

The serial I/O (SIO) section provides synchronous, double-buffered serial ports for concurrent input and output of digital data. Inputs and outputs can be handled under program control, DMA mode, or interrupt mode. The serial I/O can interface to a TDM (Time Division Multiplexed) line, serial codecs, or other DSPs (DSP32, DSP32C, DSP3210, DSP16, DSP16A) with few, if any, additional components.

The SIO is divided into three functional units, the serial input port (SIP), the serial output port (SOP), and the SIO clock generator (see Figure 9-1). There is a register, ioc, that allows the programmer to configure these functional units for various modes of operation. Registers that are accessible to the programmer are denoted by names in lower-case.

The description of the SIO is divided into five sections: SIO programming model, pin descriptions, serial input port, serial output port, and on-chip SIO clock generator. This chapter also includes a section on special purpose SIO flags and error condition reporting. The remainder of the SIO section describes programmed, DMA, and interrupt modes of handling serial I/O activity.

9.1.1 SIO Programming Model

The programming model for the SIO consists of three MMIO registers: ibuf (MMIO[0x0404]), obuf (MMIO[0x0408]), and ioc (MMIO[0x040C]). The MMIO address corresponds to the 16 LSB's of the 32-bit address. The address is given for both big and little endian byte orderings. The byte order is specified by pcw[8]. The upper 16-bits are a function of the memory mode configured in the pcw register. The register programming model for the SIO is pictured below:

Register Name	Address Big Endian	Address Little Endian	Register Bits 31 24 23 16 15 8 7 0	
ibuf	0x0404*	0x0404*		R
obuf	0x0408†	0x0408†		W
ioc	0x040C	0x040C		R/W

* Shows 32-bit read address.
† Shows 32-bit write address.

The ioc (input/output control) register is a 32-bit read-write register that configures the serial input port, the serial output port, and the on-chip SIO clock generator. The position, mnemonic, encoding, and the definition of each bit field of the ioc register are described in Table 9-1. The serial input buffer, ibuf, may be read as an 8-bit, 16-bit, or 32-bit register. Referencing ibuf is a function of bit mode (least significant bit first/most significant bit first), byte ordering, and data size. When ibuf is used as an 8-bit or 16-bit register, valid data is read as follows:

Mode	Byte Ordering	Data Size	Register Bits	Address
LSB	BE	8	31—24	0x0404
	LE	8	31—24	0x0407
MSB	BE	8	7—0	0x0407
	LE	8	7—0	0x0404
LSB	BE	16	31—16	0x0404
	LE	16	31—16	0x0406
MSB	BE	16	15—0	0x0406
	LE	16	15—0	0x0404

Referencing obuf is a function of byte ordering and data size. When obuf is used an an 8-bit or 16-bit register, valid data is written as follows:

Mode	Byte Ordering	Data Size	Register Bits	Address
MSB/MSB	BE	8	15—0	0x040a
	LE	8	15—0	0x0408
LSB/MSB	BE	8	7—0	0x040b
	LE	8	7—0	0x0408

The serial output buffer, obuf, may be written as an 8-bit, 16-bit, or 32-bit register. In the remainder of this chapter, the mnemonics ibuf, obuf, and ioc will be used exclusively and not the MMIO address. These registers may be read or written using any instruction supporting memory references. The DMA controller uses the ibuf and obuf as source and destination registers, respectively, for DMA transfers. Reset clears the ioc register to all zeros, but has no effect on the ibuf and obuf registers.

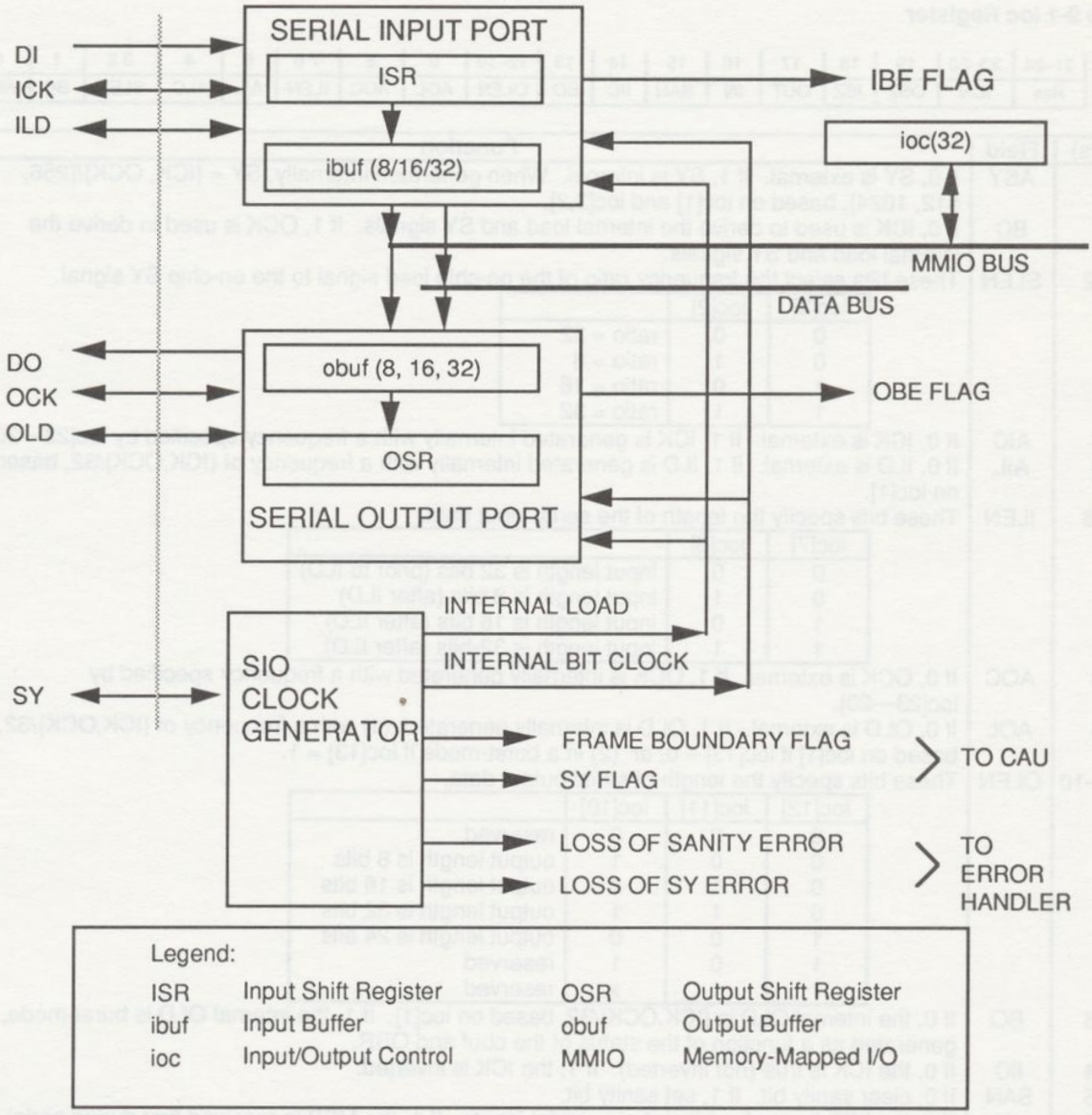


Figure 9-1 SIO Block Diagram

Table 9-1 ioc Register

Bit	31–24	23–20	19	18	17	16	15	14	13	12–10	9	8	7 6	5	4	3 2	1	0
Field	Res	ICN	OSZ	ISZ	OUT	IN	SAN	IIC	BO	OLEN	AOL	AOC	ILEN	AIL	ALC	SLEN	BC	ASY

Bit(s)	Field	Function
0	ASY	If 0, SY is external. If 1, SY is internal. When generated internally, SY = {ICK, OCK}/{256, 512, 1024}, based on ioc[1] and ioc[3,2].
1	BC	If 0, ICK is used to derive the internal load and SY signals. If 1, OCK is used to derive the internal load and SY signals.
3,2	SLEN	These bits select the frequency ratio of the on-chip load signal to the on-chip SY signal.
4	AIC	If 0, ICK is external. If 1, ICK is generated internally with a frequency specified by ioc[23—20].
5	AIL	If 0, ILD is external. If 1, ILD is generated internally with a frequency of {ICK,OCK}/32, based on ioc[1].
7,6	ILEN	These bits specify the length of the serial input data.
8	AOC	If 0, OCK is external. If 1, OCK is internally generated with a frequency specified by ioc[23—20].
9	AOL	If 0, OLD is external. If 1, OLD is internally generated: (1) with a frequency of {ICK,OCK}/32, based on ioc[1] if ioc[13] = 0, or (2) in a burst-mode if ioc[13] = 1.
12—10	OLEN	These bits specify the length of serial output data.
13	BO	If 0, the internal OLD is {ICK,OCK}/32, based on ioc[1]. If 1, the internal OLD is burst-mode, generated as a function of the status of the obuf and OSR.
14	IIC	If 0, the ICK is true (not inverted). If 1, the ICK is inverted.
15	SAN	If 0, clear sanity bit. If 1, set sanity bit.
16	IN	If 0, the LSB is received first during serial inputs. If 1, the MSB is received first during serial inputs.
17	OUT	If 0, the LSB is transmitted first during serial outputs. If 1, the MSB is transmitted first during serial outputs (can't be used with 24 bit output length).
18	ISZ	If 0, the input DMA transfer size is 32-bits. If 1, the input DMA size is specified by ILEN.
19	OSZ	If 0, the output DMA transfer size is 32-bits. If 1, the output DMA size is specified by OLEN.
23—20	ICN	These bits specify the divide-down rate of the internal clock generator. The internal clock generated has an additional divide-down by four, i.e., if bits 23—20 represent N, the clock rate is N*4. If ICN=0, the active clock generator is disabled.
31—24		Reserved; Read as zero.

ioc[3]	ioc[2]	
0	0	ratio = 32
0	1	ratio = 8
1	0	ratio = 16
1	1	ratio = 32

ioc[7]	ioc[6]	
0	0	input length is 32 bits (prior to ILD)
0	1	input length is 8 bits (after ILD)
1	0	input length is 16 bits (after ILD)
1	1	input length is 32-bits (after ILD)

ioc[12]	ioc[11]	ioc[10]	
0	0	0	reserved
0	0	1	output length is 8 bits
0	1	0	output length is 16 bits
0	1	1	output length is 32 bits
1	0	0	output length is 24 bits
1	0	1	reserved
1	1	x	reserved

Table 9-2 Pin Descriptions

Pin	Name	Description
DI	Data Input	Input for serial digital data. Data is sampled with the rising edge of ICK if IIC = 0, or the falling edge if IIC = 1.
ICK	Input Clock	Clock for serial digital input data. Serial input port activity is based on ICK. Depending on the ioc setting, this signal may be either an input (external mode) or an output (internal mode).
ILD	Input Load	A high-to-low transition on this pin synchronizes the loading of the input buffer from the input shift register. Depending on the ioc setting, the signal can be either an input (external mode) or an output (internal mode).
DO	Data Output	Output for serial digital data. Data is shifted with the rising edge of OCK.
OCK	Output Clock	Clock for serial digital output data. Serial output port activity is based on the rising edge of OCK. Depending on the ioc setting, this signal may be either an input (external mode) or an output (internal mode).
OLD	Output Load	A high-to-low transition on this pin causes the loading of the output shift register from the output buffer. Depending on the ioc setting, the signal can be either an input (external mode) or an output (internal mode).
SY	Synchronization	Provides frame synchronization either to the DSP3210 from peripheral components (external mode) or from the DSP3210 to peripheral components (internal mode), depending on the ioc setting. The logical state of this pin may be tested by CA instructions (see Section 4.3 CA Instructions).

9.1.2 Pin Descriptions

The SIO has seven pins. Three pins are associated with the SIP, three with the SOP, and one with the SIO clock generator. Two pins in each of the input and output sections (corresponding to the bit clock and load signal) and the pin for the SIO clock generator (corresponding to the frame signal) are bidirectional. These signals are denoted as **internal** and are an output if they are generated by the DSP3210. These signals are denoted as **external** and are an input if they are supplied from an outside source. Table 9-2 is a description of the seven pins associated with the serial I/O. For each pin, the pin name, full name, and a description of its function are given.

9.1.3 Serial Input Port

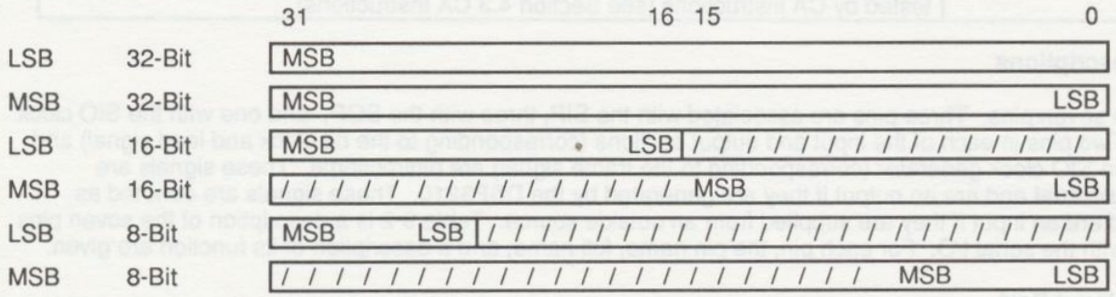
The serial input port (SIP) is used by the DSP3210 to receive serial data from external devices or data streams. The serial input port consists of the input buffer (ibuf), the input shift register (ISR), the input load shift register (ILSR), and associated logic, as shown in Figure 9-2. The SIP supports a synchronous three wire protocol: clock (ICK), load (ILD), and data (DI). An internal flag, input buffer full (IBF), is also available to indicate the status of the ibuf.

The edge of the clock used to clock data in the SIP is configurable by ioc[14], IIC. When IIC=0, the serial input port is sensitive to the rising edge of ICK. When IIC=1, ICK is inverted on-chip making the port sensitive to the falling edge of ICK. Serial data in (DI) and the load signal (ILD) are clocked into the ISR and ILSR, respectively. The load signal is delayed in the ILSR at the bit clock rate and is used to generate a signal to load the ibuf from the ISR. The number of clocks that the load signal is delayed is a function of the input length (ILEN) specified in the ioc register. When the IBUF is loaded, the input buffer full (IBF) flag is set high. When the IBUF is read by a DSP3210 program, interrupt routine, or DMA facility, the IBF flag is set low. Together, the ISR and IBUF form a double buffer so data can be shifted into the ISR while data is waiting to be read from the IBUF.

9.1.3.1 Configuring the Serial Input Port

ICK and ILD can be supplied off-chip (external mode), from an external peripheral, or on-chip (internal mode) from the on-chip SIO clock generator (see 9.1.5 SIO On-chip Clock Generator). The selections of internal or external mode for ICK, ioc[4], and ILD, ioc[5], are totally independent of each other. The internal clock generator, as configured by ioc[3-0] and ioc[23—20], generates the internal bit clock, load signal, and sync signal. The internal bit clock rate can be set for rates between CKI/4 and CKI/60 via ioc[23—20], ICN. The internal load signal is a divide by 32 of either ICK or OCK, based on ioc[1]. Note that with a continuous bit clock, the internal load will occur every 32 periods of the bit clock. The ICK can be configured to be true or inverted by ioc[14], IIC. When ICK is true, the SIP is sensitive to the rising edge of ICK. When ICK is inverted, the SIP is sensitive to the falling edge of ICK.

The length of the input data can be specified to be 8, 16 or 32 bits via ioc[6,7], ILEN. If ILEN is set to 00, then the input data corresponds to the 32 bits **prior** to the ILD high-to-low transition. This mode can be used for data lengths other than 8, 16, and 32 bits by masking off the undesired bits. For the 01,10, and 11 cases, the input data corresponds to the 8, 16, or 32 bits **after** the ILD high-to-low transition, respectively. These two protocols for serial input are shown in Figure 9-3 (IIC = 0, positive edge-triggered, is shown). The order of the serial input data bits can be specified as least significant bit first, LSB, or most significant bit first, MSB, via ioc[16], IN. IN is used to control the direction of the bidirectional shift register ISR and select which bytes of the IBUF are put on the internal Data or MMIO Bus when the input buffer is read. Input data resides in the ibuf register bits as a function of LSB or MSB mode and the size specified. Valid data in the input buffer is organized as follows:



The Data Bus is used for DMA controlled transfers and the MMIO bus is used for programmed transfers.

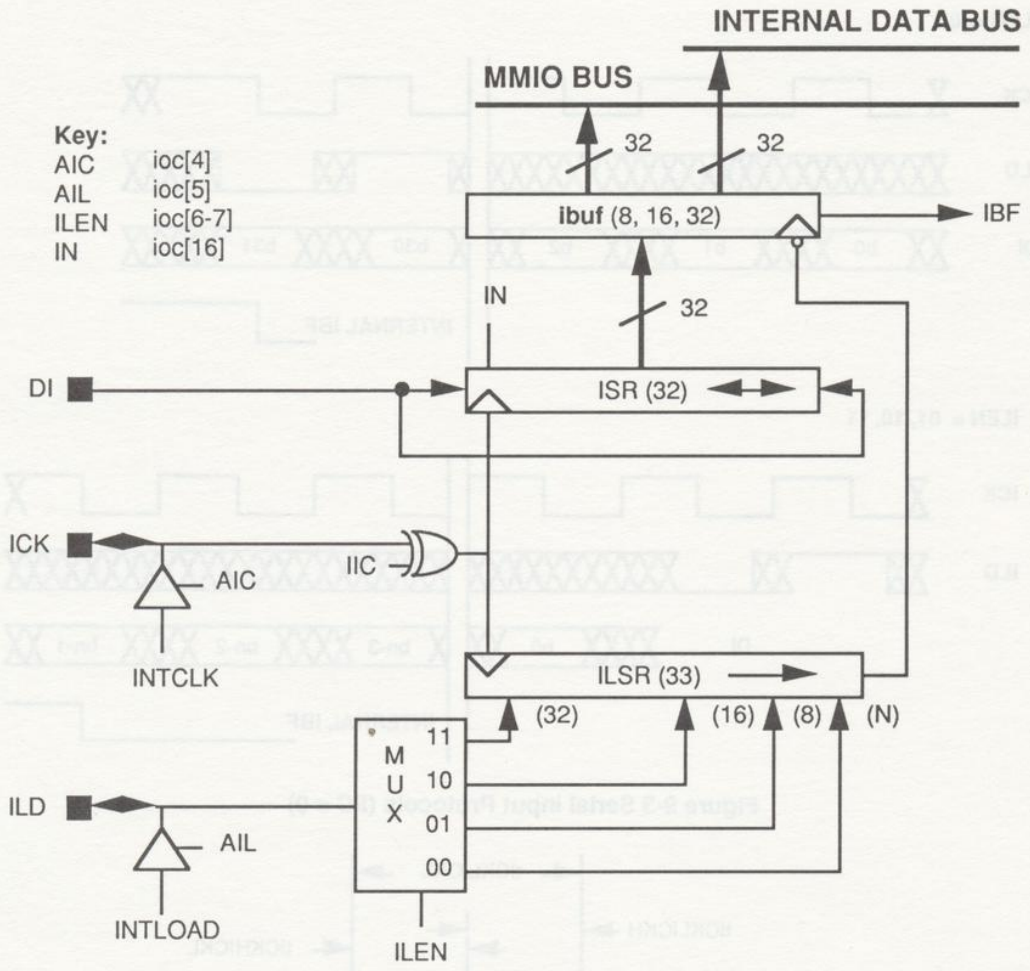


Figure 9-2 Serial Input Port

9.1.3.2 Serial Input Timing Specifications

The timing requirements and characteristics for serial input transmissions are shown in Figure 9-4 (IIC = 0, not inverted). The specifications are described as follows:

- **ICK Clock Requirements:** The clock requirements are the period of the input data bit clock ICK (tICKLICKL), the high time of ICK (tICKHICKL), and the low time of ICK (tICKLICKH).
- **ILD Requirements to Initiate an Input Transmission:** ILD is set up high before the rising edge of ICK (tILDHICKH), and is held high until after the rising edge of ICK (tICKHILDL). ILD is set up low before the next rising edge of ICK (tILDLICKH) and is held low until after that edge of ICK (tICKHILDH). Similar requirements must be met when IIC=1 (negative edge-triggered).
- **DI Requirements:** Serial input data bits 0 through n-1 must satisfy set up (tDIVICKH) and hold (tICKHDIX) times to be latched and shifted in the ISR on the rising edge of ICK. Similar requirements must be met when IIC = 1 (negative edge-triggered).

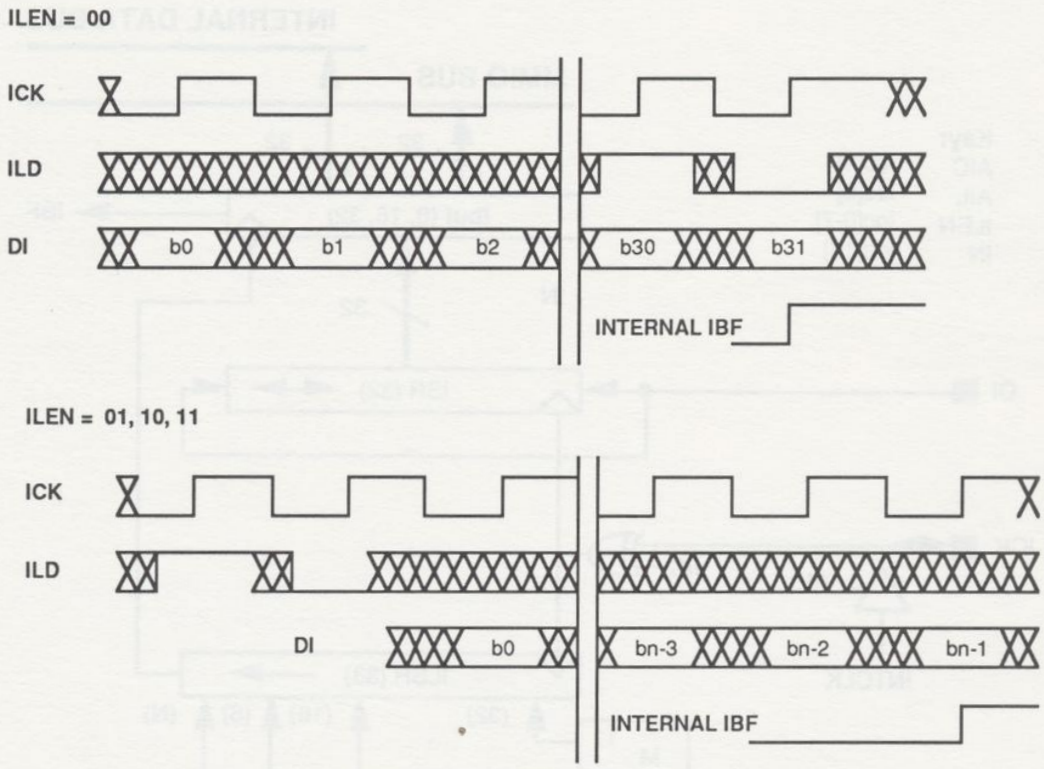


Figure 9-3 Serial Input Protocols (IIC = 0)

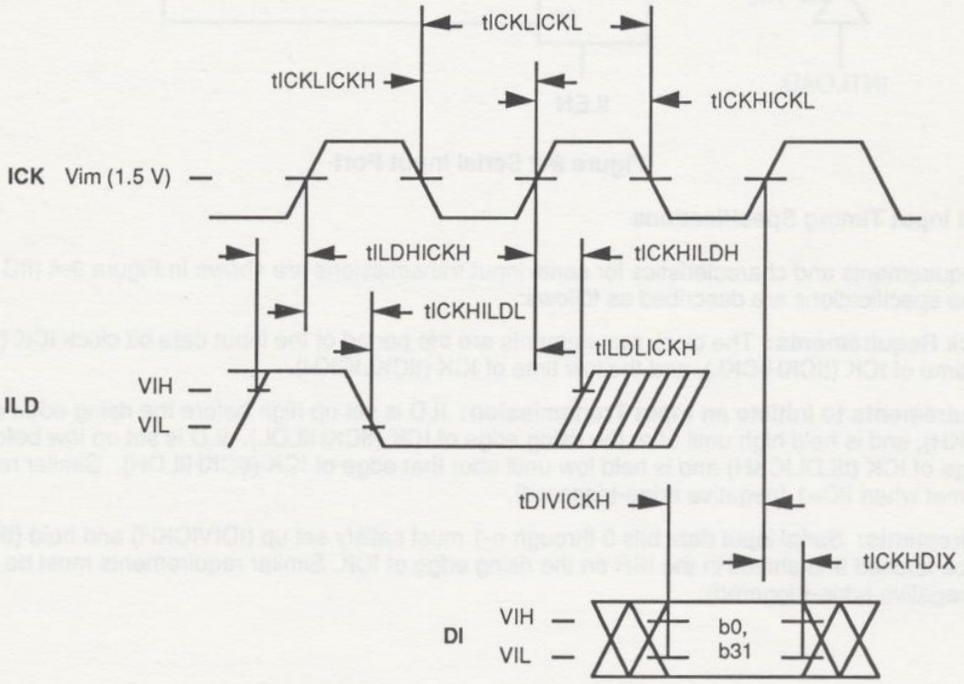


Figure 9-4 Serial Input Timing (IIC = 0)

9.1.4 Serial Output Port

The serial output port (SOP) is used to transmit serial data from the DSP3210 to external devices or data streams. The serial output port consists of the output buffer (obuf), the output shift register (OSR), the output load shift register (OLSR), and associated logic, as shown in Figure 9-5. The SOP supports a synchronous three-wire protocol: clock (OCK), load (OLD), and data (DO). An internal flag, output buffer empty (OBE), is available to indicate the status of the OBUF.

Serial output data on DO is clocked out of the output shift register (OSR) on the rising edge of the clock OCK. When the obuf is loaded by the program, the interrupt routine, or the DMA facility, the output buffer empty (OBE) flag is set low. When the data in obuf is transferred to the OSR, the OBE flag is set high. Together, the obuf and OSR form a double buffer so that data can be shifted out of the OSR while data is loaded into the obuf. On each rising edge of OCK, the load signal (OLD) is clocked and shifted in the output load shift register (OLSR). The load signal is delayed by the OLSR at the bit clock rate and is used in the generation of the load/shift signal for the OSR and the internal burst OLD (INTLOADB). If selected by ioc[13] and ioc[9], a burst OLD signal can be generated by the SIO that is active only when new data has been loaded into the obuf rather than on a periodic basis.

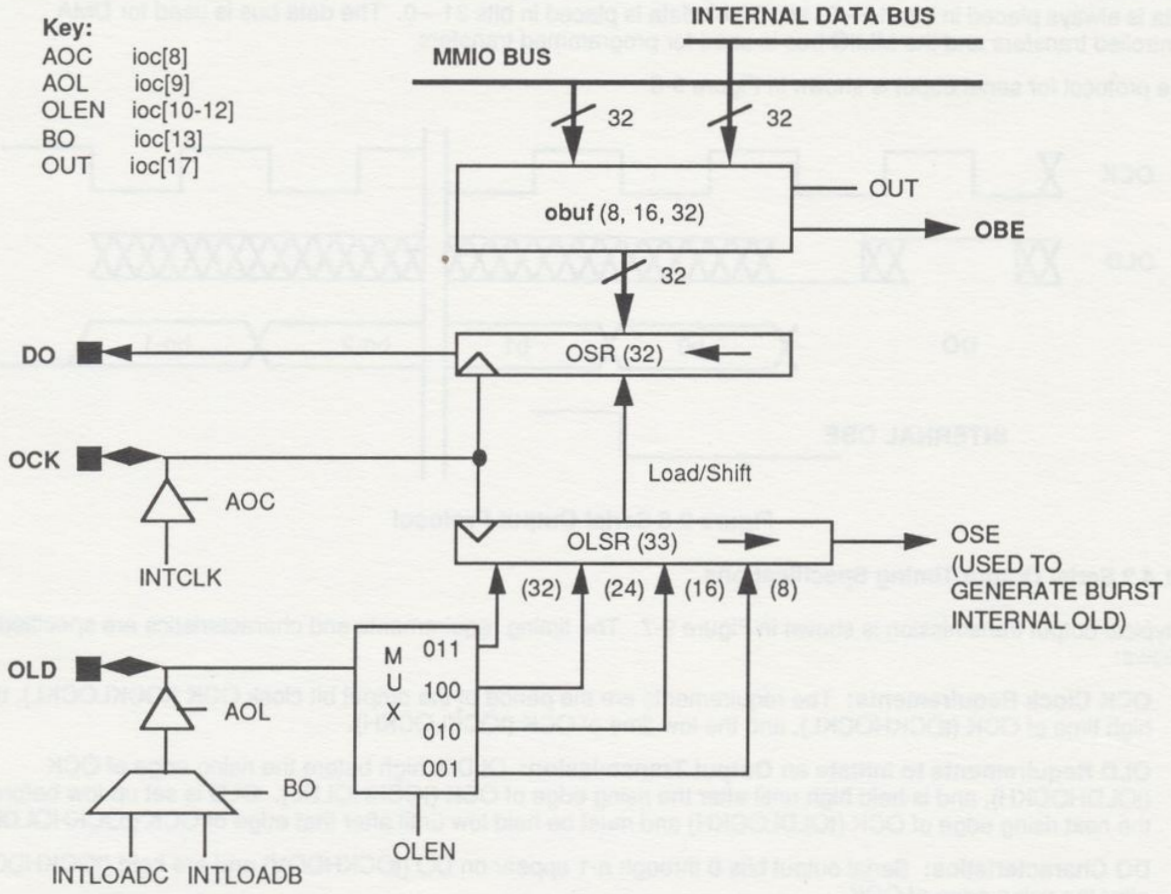


Figure 9-5 Serial Output Port

9.1.4.1 Configuring the Serial Output Port

OCK and OLD can be supplied off-chip (external mode) from an external peripheral, or on-chip (internal mode) from the on-chip SIO clock generator (see Section 9.1.5 SIO On-chip Clock Generator). The selections of internal or external mode for OCK, ioc[8], and OLD, ioc[9], are totally independent of each other. The internal clock generator, as configured by ioc[3—0] and ioc[23—20], generates the internal bit clock and load signal. The internal bit clock rate can be set for rates between CKI/4 and CKI/60 via ioc[23—20], ICN. The internal load signal can be configured in two ways: (1) continuous or (2) burst. The continuous internal load signal is a divide by 32 of either ICK or OCK, based on ioc[1]. Note that with a continuous bit clock, the internal load signal will occur every 32 bit clocks. The burst OLD signal uses the OBE flag and the OLSR output (OSE) to generate an OLD that is active only when new data has been loaded into the OBUF.

The length of output data can be specified to be 8, 16, 24, or 32 bits via ioc[12—10]. In all cases, the data corresponds to the bits after the OLD high-to-low transition. The order of the serial output bits can be specified as least significant bit first, LSB, or most significant bit first, MSB, via ioc[17], OUT. OUT is used to select which bit of the Internal Data Bus or MMIO Bus loaded into each bit of OBUF. If OUT is set to 0, bit 0 → obuf(0), bit 1 → obuf(1), ..., bit 31 → obuf(31). If OUT is set to 1, bit 0 → obuf(31), bit 1 → obuf(30), ..., bit 31 → obuf(0). MSB can not be used with an output data length of 24 bits. Eight-bit data is always placed in bits 7—0, 16-bit data is always placed in bits 15—0, and 32-bit data is placed in bits 31—0. The data bus is used for DMA controlled transfers and the MMIO bus is used for programmed transfers.

The protocol for serial ouput is shown in Figure 9-6.

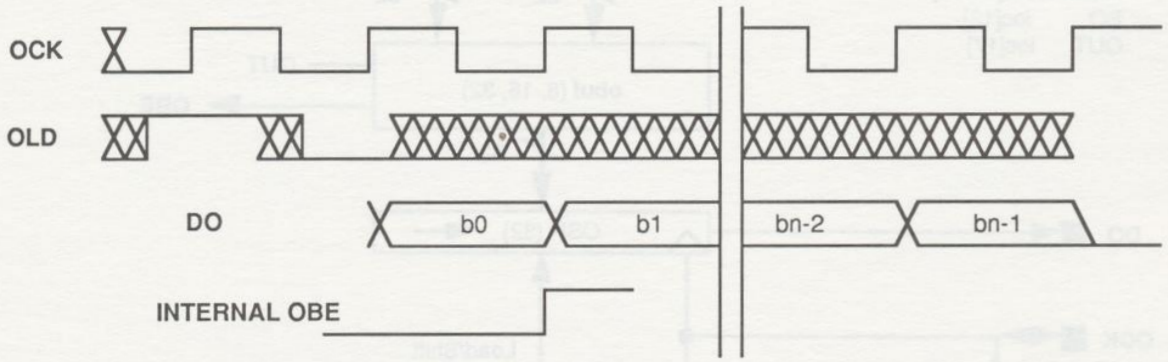


Figure 9-6 Serial Output Protocol

9.1.4.2 Serial Output Timing Specifications

A typical output transmission is shown in Figure 9-7. The timing requirements and characteristics are specified as follows:

- **OCK Clock Requirements:** The requirements are the period of the output bit clock OCK (tOCKLOCKL), the high time of OCK (tOCKHOCKL), and the low time of OCK (tOCKLOCKH).
- **OLD Requirements to Initiate an Output Transmission:** OLD is high before the rising edge of OCK (tOLDHOCKH), and is held high until after the rising edge of OCK (tOCKHOLDL). OLD is set up low before the next rising edge of OCK (tOLDLOCKH) and must be held low until after that edge of OCK (tOCKHOLDH).
- **DO Characteristics:** Serial output bits 0 through n-1 appear on DO (tOCKHDOV) and are held (tOCKHDOX) after the rising edge of OCK.

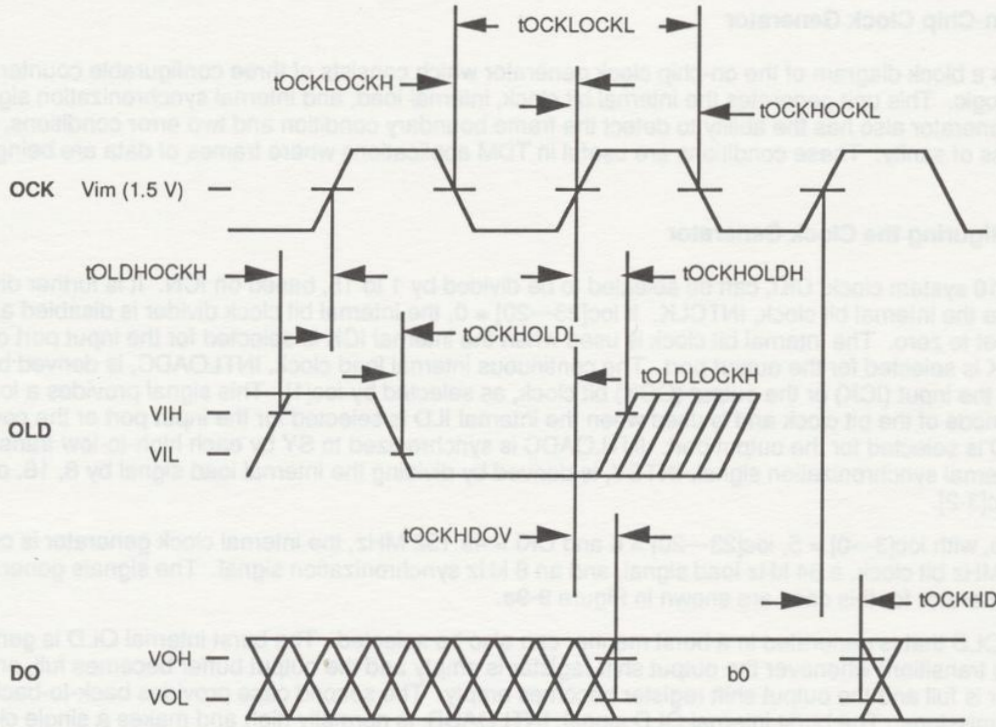


Figure 9-7 Serial Output Timing

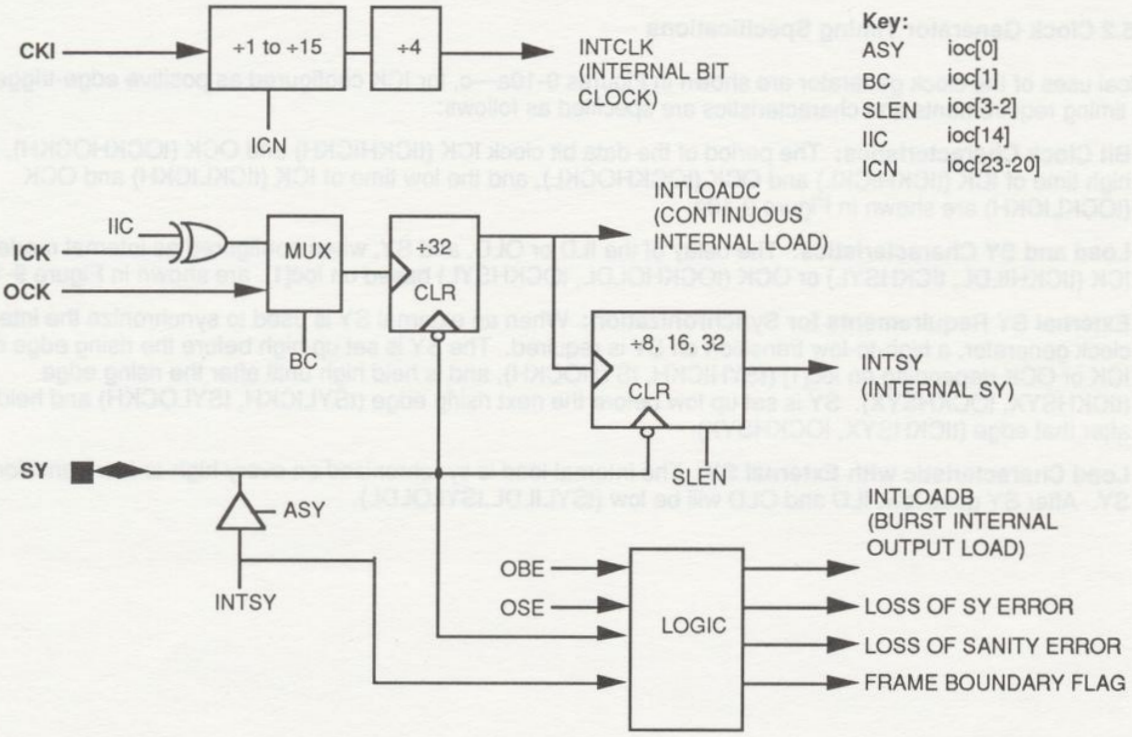


Figure 9-8 On-Chip Clock Generator

9.1.5 SIO On-Chip Clock Generator

Figure 9-8 is a block diagram of the on-chip clock generator which consists of three configurable counters and associated logic. This unit generates the internal bit clock, internal load, and internal synchronization signals. The clock generator also has the ability to detect the frame boundary condition and two error conditions, loss of sync and loss of sanity. These conditions are useful in TDM applications where frames of data are being processed.

9.1.5.1 Configuring the Clock Generator

The DSP3210 system clock, CKI, can be selected to be divided by 1 to 15, based on ICN. It is further divided by 4 to generate the internal bit clock, INTCLK. If ioc[23—20] = 0, the internal bit clock divider is disabled and INTCLK is set to zero. The internal bit clock is used when the internal ICK is selected for the input port or the internal OCK is selected for the output port. The continuous internal load clock, INTLOADC, is derived by dividing by 32 either the input (ICK) or the output (OCK) bit clock, as selected by ioc[1]. This signal provides a load signal every 32 periods of the bit clock and is used when the internal ILD is selected for the input port or the continuous internal OLD is selected for the output port. INTLOADC is synchronized to SY by each high-to-low transition of SY. The internal synchronization signal, INTSY, is derived by dividing the internal load signal by 8, 16, or 32, based on ioc[3-2].

For example, with ioc[3—0] = 5, ioc[23—20] = 6 and CKI = 49.152 MHz, the internal clock generator is configured for a 2.048 MHz bit clock, a 64 kHz load signal, and an 8 kHz synchronization signal. The signals generated by the clock generator for this case are shown in Figure 9-9a.

An internal OLD that is generated in a burst manner can also be selected. The burst internal OLD is generated (one-to-zero transition) whenever the output shift register is empty and the output buffer becomes full, or when the output buffer is full and the output shift register becomes empty. The second case provides back-to-back serial output transmissions. The burst internal OLD signal, INTLOADB, is normally high and makes a single clock cycle low pulse to indicate the start of an output. Figure 9-9b illustrates how OLD is generated with respect to the output buffer status and shift register status for the second case described above.

9.1.5.2 Clock Generator Timing Specifications

Typical uses of the clock generator are shown in Figures 9-10a—c, for ICK configured as positive edge-triggered. The timing requirements and characteristics are specified as follows:

- **Bit Clock Characteristics:** The period of the data bit clock ICK (tICKHICKH) and OCK (tOCKHOCKH), the high time of ICK (tICKHICKL) and OCK (tOCKHOCKL), and the low time of ICK (tICKLICKH) and OCK (tOCKLICKH) are shown in Figure 9-10a.
- **Load and SY Characteristics:** The delay of the ILD or OLD, and SY, when configured as internal mode from ICK (tICKHILDL, tICKHSYL) or OCK (tOCKHOLDL, tOCKHSYL) based on ioc[1] are shown in Figure 9-10b.
- **External SY Requirements for Synchronization:** When an external SY is used to synchronize the internal clock generator, a high-to-low transition on SY is required. The SY is set up high before the rising edge of ICK or OCK depending on ioc[1] (tSYHICKH, tSYHOCKH), and is held high until after the rising edge (tICKHSYX, tOCKHSYX). SY is set up low before the next rising edge (tSYLICKH, tSYLOCKH) and held low after that edge (tICKHSYX, tOCKHSYX).
- **Load Characteristic with External SY:** The internal load is synchronized on every high-to-low transition of SY. After SY goes low, ILD and OLD will be low (tSYLILDL,tSYLOLDL).

9.1.5.3 Loss of Sanity

Note: Error Condition Handling for Loss of Sanity is not supported in the DSP3210.

Loss of sanity is defined to be a high-to-low transition on SY while ioc[15], SAN, is a one. To make use of this feature, SAN should be set to one at the beginning of the program processing the frame of data, and cleared to zero before waiting for SY to be zero. If the program processing the frame takes longer that one frame interval, then the loss of sanity error is reported to the error exception handler.

9.1.5.4 Loss of Sync

Note: Error Condition Handling for Loss of Sync is not supported in the DSP3210.

Loss of sync is defined to be the occurence of an INTSY high-to-low transition (from the on-chip SIO clock generator) without a corresponding external SY high-to-low transition (from the SY pin), when SY is applied externally (ASY = 0). Note that the INTSY is synchronized (divide down counters are set to zero) by a high-to-low transition on the SY pin. For the loss of sync error to be detected, ioc[0] must be 0, and ioc[1—3] must be set so that the INTSY is the same frequency as the SY pin. If a loss of sync is detected, an error is reported to the error exception handler.

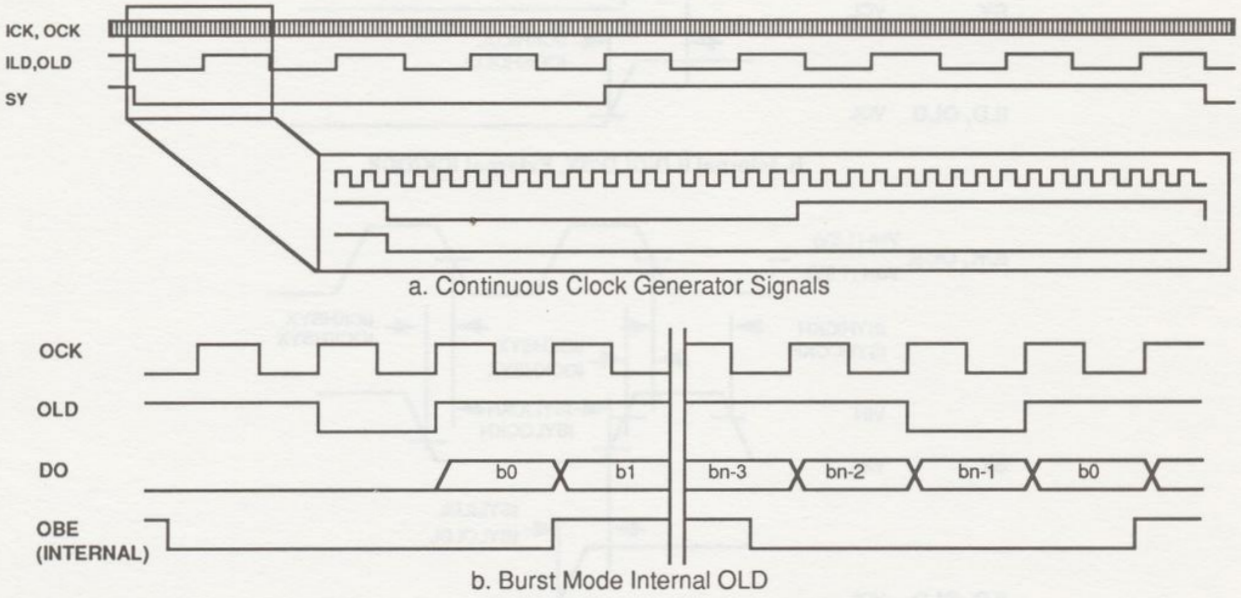
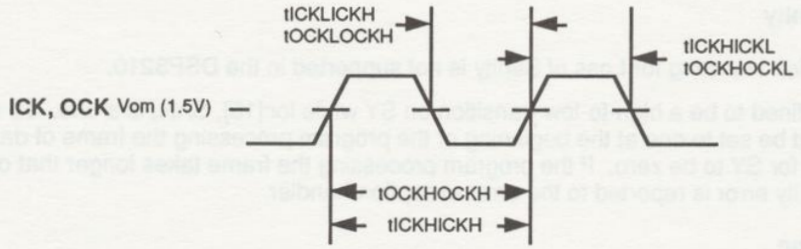
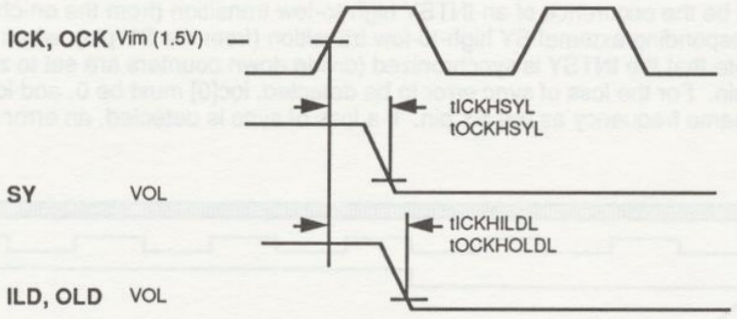


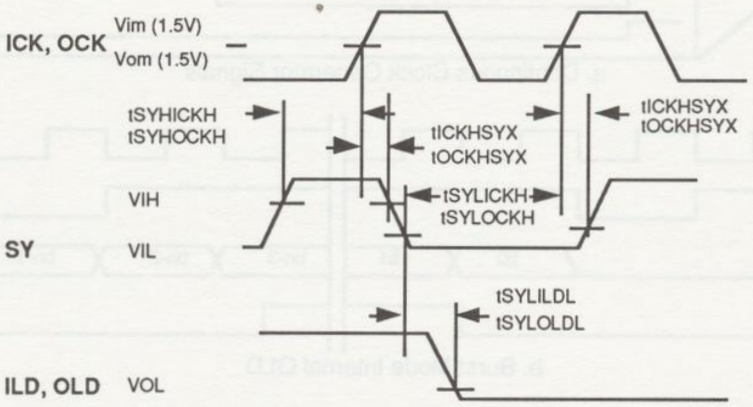
Figure 9-9 Clock Generator Signals



a. Internal ICK/OCK



b. Internal ILD/OLD/SY. External ICK/OCK



c. External SY. Internal ILD/OLD.
External or Internal ICK/OCK.

Figure 9-10 Clock Generation Timing (IIC = 0)

9.1.5.5 Frame Boundary Condition

The on-chip clock generator detects the frame boundary (fb) condition which is useful in certain TDM applications. The fb condition is defined to be the first time the output buffer is full following a high-to-low transition on SY, provided that ioc[15], SAN, is equal to zero at that time. It uniquely indicates the first channel in a TDM stream. After the SY transition is detected while SAN is zero, the fb flag will be the inverse of OBE until it is cleared to zero by setting SAN to a one. The frame boundary condition can be tested using conditional branch instructions, e.g. if(fbs)goto label (see 4.3.1 Control Arithmetic Instructions - Control Group). The frame boundary condition is usually used in conjunction with the loss of sanity error condition because toggling of the SAN bit is required for both. Figure 9-11 illustrates how the frame boundary condition is used in a TDM application.

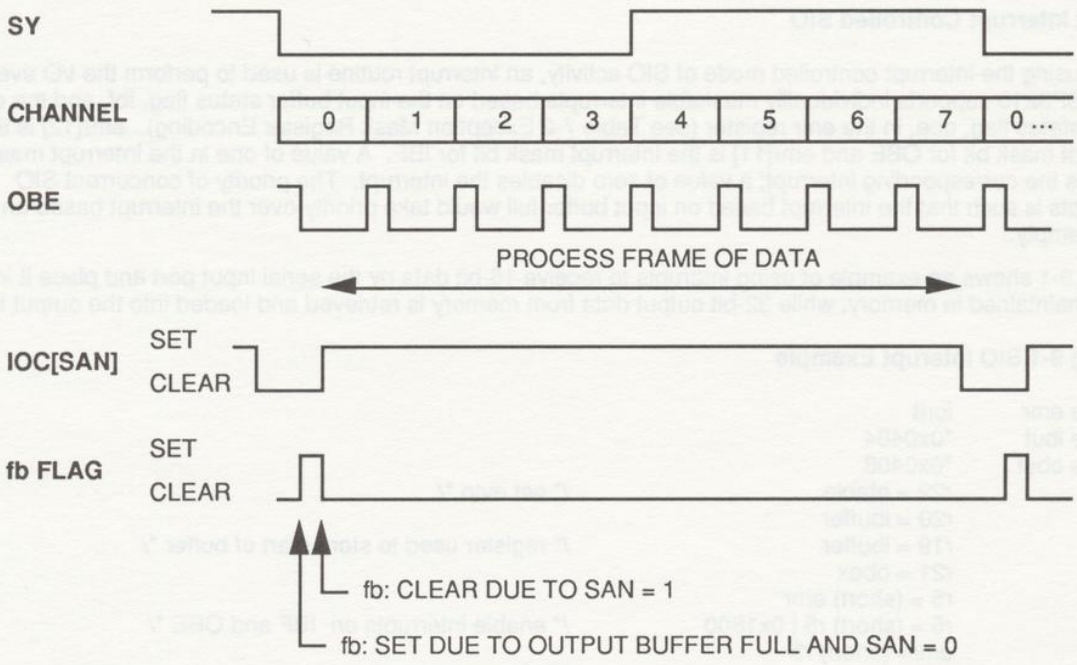


Figure 9-11 Frame Boundary Condition

9.1.6 Handling SIO Activity

This section describes three typical ways of handling SIO activity: program controlled, interrupt controlled, and DMA controlled. An example is given for each.

9.1.6.1 Program Controlled SIO

When using the program controlled mode of SIO activity, flags pertaining to the status of the input buffer, output buffer, or data stream are tested by the program. When the appropriate condition becomes true, an I/O event is initiated by the program. There are four flags available to the programmer, ibf, obe, sy, and fb. Both true and false states of these flags are directly testable by an instruction of the form:

if (IO COND) goto {rH, N, rH+N}

For more information, see 4.3.1 Control Arithmetic Instructions - Control Group.

For example, in order to process input data as it is received by the serial input port, the program must wait until the input buffer is full. This is accomplished by the following instructions:

```
waitibf:    if(ibe) pcgoto waitibf      /* conditional branch using pc relative addressing */
            nop                         /* latent instruction */
```

9.1.6.2 Interrupt Controlled SIO

When using the interrupt controlled mode of SIO activity, an interrupt routine is used to perform the I/O event. The DSP3210 supports individuallly maskable interrupts based on the input buffer status flag, ibf, and the output buffer status flag, obe, in the emr register (see Table 7-2 Exception Mask Register Encoding). emr[12] is the interrupt mask bit for OBE and emr[11] is the interrupt mask bit for IBF. A value of one in the interrupt mask bit enables the corresponding interrupt; a value of zero disables the interrupt. The priority of concurrent SIO interrupts is such that the interrupt based on input buffer full would take priority over the interrupt based on output buffer empty.

Listing 9-1 shows an example of using interupts to receive 16-bit data by the serial input port and place it in a FIFO maintained in memory, while 32-bit output data from memory is retrieved and loaded into the output buffer.

Listing 9-1 SIO Interupt Example

```
#define emr     ior8
#define ibuf    *0x0404
#define obuf    *0x0408
                r22 = etable                    /* set evtp */
                r20 = ibuffer
                r19 = ibuffer                   /* register used to store start of buffer */
                r21 = obox
                r5 = (short) emr
                r5 = (short) r5 | 0x1800        /* enable interrupts on  IBF and OBE */
                emr= (short) r5
main:           .                               /* program body */
                .
                .
                pcgoto main
ibuffer:        N*int                           /* FIFO storage in memory  */
obox:           float
etable:         22*nop                          /* First eleven entries of evt */
                pcgoto ibfisr                   /* branch to IBF interrupt service routine */
                nop
                pcgoto obeisr                   /* branch to IBF interrupt service routine */
                nop
                8*nop                           /* Remainder of evt */
ibfisr:         r1= (short) ibuf
                nop
                *r20++ = (short) r1             /* write ibuf to memory */
                r20 - r21                       /* compare r20 to end of buffer  */
                if (mi) r20 = r19               /* reload starting address if at end */
                ireturn
obeisr:         r1=*r21
                nop
                obuf=r1                         /* load obuf from r1 */
                ireturn
```

9.1.6.3 DMA Controlled SIO

The DSP3210 includes a DMA option for transferring data between the ibuf, obuf, and memory without program intervention. The serial DMA option is particularly useful for multichannel applications in which input and output buffers for an entire frame of samples are established in memory.

The DSP3210 contains a DMA controller block to perform the necessary address generation and control for the DMA process (see Section 9.2 DMA Controller). The DMA controller consists of two channels, one for input DMA and one for output DMA. The registers to configure and enable DMA activity are MMIO registers and are described fully in Section 9.2. All serial I/O DMA activity is based on either the IBF flag or the OBE flag. Based on one or both of these flags, an input DMA will occur, an output DMA will occur, or both an input and an output DMA will occur. The last type will perform both operations, input DMA and then output DMA, in one DMA cycle, where a DMA cycle is equal to one instruction cycle.

To buffer a frame of serial input data in DSP3210 memory, the following registers need to be setup:

icnt	transfer count for input DMA
idp	input dma pointer
dmac	bits 4-7 define the mode of input DMA

For a buffer of 128, 16-bit samples, the icnt register should be configured for 0x00fe0000 (one less than the number of transfers desired times the data size in bytes). The idp should be set for the base address of the buffer which in this case must be aligned for 16-bit data (bit 0 must be 0). After the icnt and idp registers have been initialized, the dmac is loaded with the desired configuration. Program Listing 9-2 shows the code necessary to initialize the input channel of the DMA controller and enable input DMA. An interrupt can be taken when the input DMA transfer is complete by unmasking the IFRM bit of the emr register and supplying an appropriate interrupt routine.

Listing 9-2 Input DMA Example

```
#define icnt    *0x428
#define idp     *0x420
#define dmac    *0x430

        r1= 0xfe        /* icnt for 128, 16-bit samples */
        icnt=r1
        r1=ibuffer      /* starting address of input buffer */
        idp=r1
        r1=0x30         /* enable input DMA, and stop when transfer count is reached */
        dmac=r1

ibuffer: 128*int        /* input DMA buffer */
```

9.2 DMA Controller (DMAC)

The DMA controller is a two channel Direct Memory Access (DMA) control unit. It is used in conjunction with the Serial I/O to buffer input and output data streams in DSP3210 memory, internal or external. The DMA transfers occur between memory and the SIO ports without processor intervention, i.e., using cycle-stealing where the DSP3210 waits during a DMA transfer (see Section 7.3 Direct Memory Access).

9.2.1 DMAC Programming Model

The register programming model for the DMA controller consists of five MMIO registers. The idp and the odp are the input DMA and the output DMA pointer registers, respectively. Each is 32-bits and therefore can point to any location in the DSP3210 address space. When read, the idp and odp return the value of the current pointer register. The icnt and ocnt registers define the transfer count for input and output DMA frames, respectively. A DMA frame is considered to be the number of DMA transfers to be performed. The transfer count is the number of bytes to be transferred. **The transfer count is specified in bytes and is set to one transfer less than the frame size;** e.g., for an output frame size of 64 and a transfer size of 16-bits the ocnt register is set for 126 or (64 – 1) * 2. When read, the icnt and ocnt return the value of the current transfer count register. The dmac register is used to configure the operating modes of both DMA controllers. The dmac register is set to zero on reset. The encoding and functional definition of each bit field of the dmac register are described in Table 9-3. The address is given for both big and little endian byte orderings. The byte order is specified by pcw[8]. The register programming model for the DMAC is pictured below:

Register Name	Address Big Endian	Address Litttle Endian	Register Bits 31 16 15 8 7 0	
idp	0x0420	0x0420		R/W
odp	0x0429	0x0424		R/W
icnt	0x042a	0x0428		R/W
ocnt	0x042e	0x042C		R/W
dmac	0x0433	0x0430		R/W

9.2.2 DMAC Functional Description

Each DMA controller consists of a base address register, a transfer count register, an adder, a counter, and a comparator (see Figure 9-12). A DMA channel can be configured to operate in one of two primary modes: stop on transfer complete mode or reload mode. The stop on transfer complete mode is used to transfer a single frame of data, and the reload mode is normally used when transferring continuous frames of data. In the reload mode, the base address register and the transfer count register operate in a double buffered manner. The register currently being used is called the **current** register, and the register to be used next is called the **next** register.

In the stop on transfer complete mode (ORS = 0 or IRS = 0), when the processor loads the idp, odp, icnt, or ocnt registers, both the **next** register and the **current** register are written, and the transfer counter is set to zero. The address of the current location being pointed to is generated by adding the counter value to the current DMA pointer register. The comparator is used to compare the counter output to the current transfer count. When the counter equals the current transfer count the enable/disable bit (ODE or IDE) is set to zero, and the counter is set to zero.

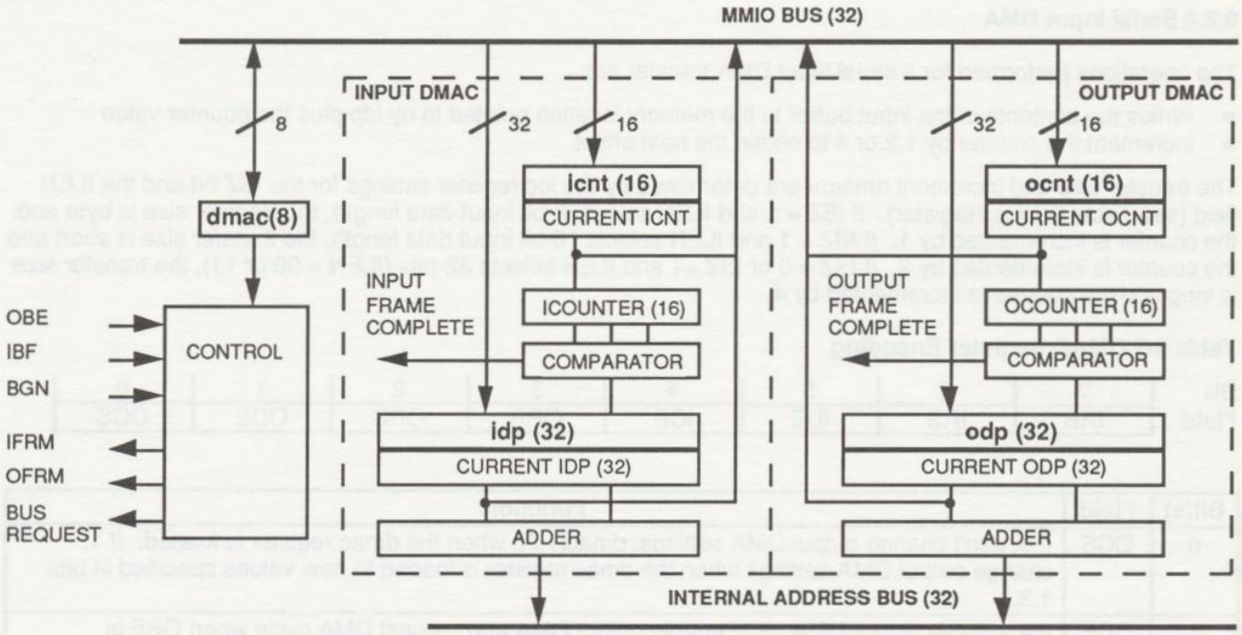


Figure 9-12 DMAC Block Diagram

In the reload mode (ORS = 1 or IRS = 1), when the processor loads the idp, odp, icnt, or ocnt registers, only the **next** register is affected. When the counter equals the current transfer count, the current pointer register and current transfer count registers are loaded from the next pointer register and next transfer count registers, respectively, and the transfer counter is set to zero. The enable/disable bit is unaffected. When the first frame is being set up, the dmac register should be configured such that DMA is disabled and the channel is in the stop on transfer complete mode.

A mode is provided so that DMA cycles are requested only when the DSP3210 has ownership of its bus interface. This is useful when the DSP3210 is buffering data to or from a shared external memory, because it guarantees that the DMA cycle will not incur the overhead of waiting to gain ownership of the bus. This mode is enabled with the ORB for output DMA and IRB for input DMA. It operates by requesting the bus interface when a DMA operation has been requested, but not performing the DMA cycle until the bus has been granted.

Note: When a DMA controller is enabled, the mode of operation, as specified by the dmac register, and the size of the transfer, as specified by the ioc register, should not be changed.

9.2.3 Serial Output DMA

The operations performed for a serial output DMA transfer are:

- write the contents of the memory location pointed to by odp plus the counter value to the output buffer
- increment the counter by 1,2,or 4 to obtain the next offset.

The transfer size and increment amount are determined by the ioc register settings for the OSZ bit and the OLEN field (see Table 9-1 ioc Register). If OSZ = 1 and OLEN selects 8-bit input data length, the transfer size is byte and the counter is incremented by one. If OSZ=1 and OLEN selects 16-bit input data length, the transfer size is short and the counter is incremented by 2. If OSZ = 0 or OSZ = 1 and OLEN selects 32-bits (OLEN = 011 or 100), the transfer size is long and the counter is incremented by 4.

9.2.4 Serial Input DMA

The operations performed for a serial input DMA transfer are:

- Writes the contents of the input buffer to the memory location pointed to by idp plus the counter value
- Increment the counter by 1,2,or 4 to obtain the next offset

The transfer size and increment amount are determined by the ioc register settings for the ISZ bit and the ILEN field (see Table 9-1 ioc Register). If ISZ = 1 and ILEN selects 8-bit input data length, the transfer size is byte and the counter is incremented by 1. If ISZ = 1 and ILEN selects 16-bit input data length, the transfer size is short and the counter is incremented by 2. If ISZ = 0 or ISZ =1 and ILEN selects 32-bits (ILEN = 00 or 11), the transfer size is long and the counter is incremented by 4.

Table 9-3 DMAC Register Encoding

Bit	7	6	5	4	3	2	1	0
Field	IRB	IRS	IDE	ICS	ORB	ORS	ODE	OCS

Bit(s)	Field	Function
0	OCS	If 0, don't change output DMA settings, dmac[1,2], when the dmac register is loaded. If 1, change output DMA settings when the dmac register is loaded to new values specified in bits 1,2.
1	ODE	If 0, disable output DMA. If 1, enable output DMA and request DMA cycle when OBE is asserted, obuf is empty.
2	ORS	If 0, stop output DMA when the number of DMA transfers (OCOUNTER) reaches the number specified in OCNT. If 1, reload the current output DMA pointer and the current output transfer count registers, ODP and OCNT, respectively, when the number of DMA transfers (OCOUNTER) reaches the number specified in OCNT and keep output DMA enabled.
3	ORB	If 0, request output DMA cycles based on OBE and dmac[1]. If 1, request output DMA cycles based on OBE, dmac[1], **and** if the DSP3210 has ownership of the bus interface. If the DSP3210 is not the current bus master, the BRN is asserted. When the bus is granted, the DMA cycle is requested.
4	ICS	If 0, don't change input DMA settings, dmac[4,5], when the dmac register is loaded. If 1, change input DMA settings when the dmac register is loaded to new values specified in bits 4,5.
5	IDE	If 0, disable input DMA. If 1, enable input DMA and request DMA cycle when IBF is aserted, ibuf is full.
6	IRS	If 0, stop input DMA when the number of DMA transfers (ICOUNTER) reaches the number specified in ICNT. If 1, reload the current input DMA pointer and the current input transfer count registers, IDP and ICNT, respectively, when the number of DMA transfers (ICOUNTER) reaches the number specified in ICNT and keep output DMA enabled.
7	IRB	If 0, request input DMA cycles based on IBF and dmac[4]. If 1, request DMA cycles based on IBF, dmac[4], **and** if the DSP3210 has ownership of the bus interface. If the DSP3210 is not the current bus master, the BRN is asserted. When the bus is granted, the DMA cycle is requested.

9.3 Timer

The timer is a programmable 32-bit counter and associated control register and logic that can be used for interval timing, rate generation, event counting, or waveform generation. The input to the timer can be derived from the DSP3210 clock or come from an external source. The output of the timer can generate a maskable interrupt or be selected as an output of the chip to drive external hardware. The count-down timer can be configured to count to zero once, or to count continuously by automatically reloading the counter with its initial value when it reaches zero.

9.3.1 Timer Programming Model

Access to the timer is provided by reading and writing the **tcon** and **timer** MMIO registers. When the **timer** is written, the initial count register and counter are loaded. When the timer is read, the current counter value is read. The count value may be read or changed at any time during operation. The tcon register is used to configure and control the timer. Table 9-4 describes the function of each bit of the tcon register. On reset, all bits of the tcon register are set to zero and the timer is set to all 1s. The address for the timer registers is given for big and little endian byte ordering. The byte ordering is specified by pcw[8]. The register programming model for the timer is:

Register Name	Address Big Endian	Address Litttle Endian	Register Bits	
			31 ... 7 ... 0	
tcon	0x0413	0x0410		R/W
timer	0x0414	0x0414		R/W

9.3.2 Timer Functional Description

The timer consists of an 8-bit control register (tcon), the initial count register, the counter, and associated control logic (source selection, counter control, and output control), see Figure 9-13. The source selection logic is used to select the clock source for the counter and is based on the value of the tcon register. The source can be a divide down of the DSP system clock, CKI, or the source can be the external signal applied to the pin BIO0, Bit I/O 0. Divide-down rates for CKI of 2, and 4 can be selected. The true or complement of the external source, BIO0, can be selected. The maximum frequency of an external source is CKI/2. The source selection logic detects zero-to-one transitions of the selected clock source.

The counter control logic controls the loading and decrementing of the counter. The counter is loaded when the timer is written, and when the count reaches zero if tcon[1] = 1 (automatic reload of counter). If the counter is loaded when it is enabled, the loading of the counter takes preference over decrementing. The counter is decremented by one for each zero-to-one transition of the clock source detected by the source selection logic. At any time, the counter can be stopped and the count held by setting tcon[0] to a zero. Setting tcon[0] to a one at this time will continue the count-down operation from the current count.

The output control logic controls the type of output signal generated by the counter. Whenever the count reaches zero, an interrupt request is generated, the state of a toggle flip-flop changes state, and a pulse, one CKI period wide, is generated. In order for the DSP to respond to the interrupt, the timer interrupt bit in the emr register, emr[9], must be set to one (unmasked) (see Table 7-2 emr Register Encoding). Depending on the setting of tcon[2], either the output of the toggle flip-flop or the pulse will be available to be observed external to the DSP via pin BIO1. For this, pin BIO1 has to be configured as an output and tcon[4] has to be set to one. If the pulse mode is selected (tcon[2] = 0), the pulse can be active high or active low, depending on tcon[3]. The active high signal is normally low, and when the count reaches zero it will pulse **high** for one clock period. The active low signal is normally high, and when the count reaches zero it will pulse **low** for one clock period.

Use of the timer usually starts with loading the initial count register and the counter. This is done by loading the **timer** register. After loading the counter, the count-down operation is enabled by setting tcon[0] to a one. When the count reaches zero and if tcon[1] is set to a one, the counter is reloaded automatically from the initial count register, without the need for program intervention. The period of the count is N+1, where N is the value loaded in the counter. Alternatively, when the count reaches zero and if tcon[1] is set to a zero, the counter stops. In this case, the counting interval is N, where N is the value loaded into the counter.

By setting the counter to zero and configuring it to reload continuously, it is possible to generate a clock, observable externally at the BIO1 pin, which has either a CKI/2 or a CKI/4 rate. For example, a setting of TCON to 0x13 will result in an external clock with a rate of CKI/2 and 50% duty cycle; a setting of TCON to 0x17 generates an external clock with a rate of CKI/4 and 50% duty cycle; a setting of TCON to 0x33 generates an external clock with a rate of CKI/4 and 75% duty cycle (3 CKI periods high, 1 CKI period low); and a setting of TCON to 0x3b will give an external clock with a rate of CKI/4 and 25% duty cycle (1 CKI period high, 3 CKI periods low).

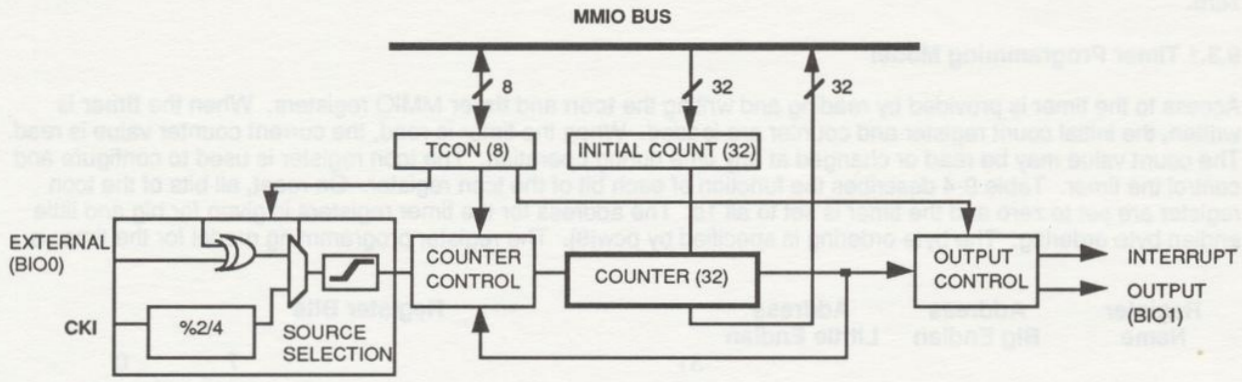


Figure 9-13 Timer Block Diagram

Table 9-4 TCON Register Encoding

Bit	7 6 5	4	3	2	1	0
Field	SRC	OUT	H/LN	T/PN	R/SN	E/HN

Bit(s)	Field	Function
0	E/HN	Hold/enable count-down operation of counter. 0 - Hold the count. 1 - Enable the count-down operation of counter.
1	R/SN	Select between continuous and one-time operation. 0 - Stop counting when the count reaches zero. 1 - Automatic reload of the counter from the initial count register when the count reaches zero.
2	T/PN	Select pulse or toggle operation of the output when the count reaches zero. 0 - For the external output, a pulse is generated when the count reaches zero. 1 - For the external output, the output is the output of a toggle flip-flop which is toggled when the count reaches zero.
3	H/LN	When tcon[2] is set to zero, pulse, select low active or high active pulse. 0 - When tcon[2] is set to zero, the pulse is one CKI clock period wide and is low active (1->0->1). 1 - When tcon[2] is set to zero, the pulse is one CKI clock period wide and is active high (0->1->0).
4	OUT	If the BIO1 pin is configured as an output, 0 - The BIO1 pin has the value specified by the BIO1 field in the BIO register. 1 - The BIO1 pin is selected to be the timer output.
5-7	SRC	Selects the clock source for the timer. tcon 7,6,5 000 Counter clock is CKI/2 001 Counter clock is CKI/4 010 Reserved 011 Reserved 100 Counter clock is from external source(pin BIO0) - rising edge 101 Counter clock is from external source (pin BIO0) - falling edge 11x Reserved

9.3.3 Timer Timing Specifications

When BIO0 or BIO1 are selected to be used as the timer clock source or the timer output, respectively, the following timing specifications apply. BIO0 is an asynchronous input and is synchronized to the DSP3210 by an internal version of CKI. If an input makes a transition during the sample window, the level recognized by the processor is not predictable; however, the processor always resolves the latched level to either a logic one or zero before using it. Figure 9-14 shows the sampling window for BIO0 and the timing relationship between BIO1 and CKI. BIO1 changes on the falling edge of CKI. The specifications for BIO0 when it is asynchronous to the DSP3210 are that the period (tBIO0HBIO0H) must be greater than twice the CKI period, and the high or low time (tBIO0HBIO0L and tBIO0LBIO0H) must be greater than one CKI period.

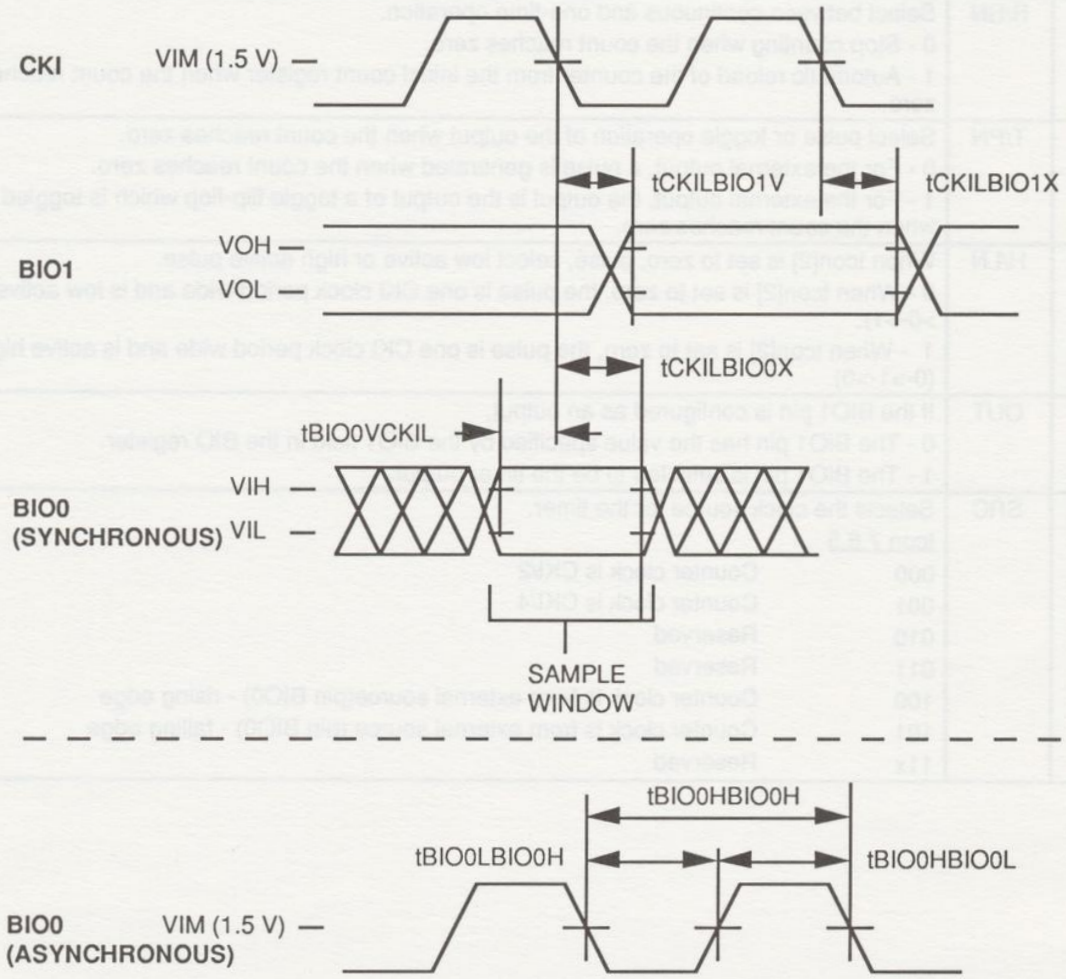


Figure 9-14 Timer Timing Specifications

9.4 Bit I/O (BIO)

The BIO is a general-purpose 8-bit input/output port. It includes features that make it suitable for board-level status signal generation and control signal testing by the DSP3210. The BIO interface consists of 8 I/O lines, any of which can be independently configured as an input or an output. Outputs can be written with a 1 or a 0, toggled, or left unchanged. Inputs can be directly read and loaded into a CAU register and then tested.

9.4.1 BIO Programming Model

Access to the BIO is provided by reading and writing the **bioc** and **bio** MMIO registers using CA Data Move Instructions (e.g. *0x0418 = (byte) r1, r1 = (short) *0x041C, ...). The bioc register is used to configure each bit of the BIO as an input or an output. Table 9-5 describes the function of each bit of the bioc register. The bio register is used to alter the state of bits configured as outputs or to determine the logical state of BIO inputs. Table 9-6 describes the function of each bit of the bio register. On reset, all bits of the bioc register and output register bits of the bio register are set to zero. The register programming model for the BIO is as follows:

Register Name	Address Big Endian	Address Litttle Endian	Register Bits
		31	16 15 8 7 0
bioc	0x041b	0x0418	R/W
bio	0x041e	0x041C	R/W

Table 9-5 bioc Register Encoding

Bit	7	6	5	4	3	2	1	0
Field	B7	B6	B5	B4	B3	B2	B1	B0

Each bit in the bioc register corresponds to one of the pins in the BIO port, B0—BIO0, B1—BIO1, ..., B7—BIO7. A value of 0 in the bit indicates that the associated pin is configured as an input; a value of 1 indicates that the pin is configured as an output. On reset all bits are set to zero (input).

Table 9-6 BIO Register Encodings (Writing and Reading)

Bit	15 14	13 12	11 10	9 8	7 6	5 4	3 2	0 1
Field	BF7	BF6	BF5	BF4	BF3	BF2	BF1	BF0

Each pair of bits (field) in the bio register corresponds to one of the bio pins in the BIO port, BF0-BIO0, BF1—BIO1, ..., BF7—BIO7. When the bio register is written, the value of each output register bit is changed according to the following encoding of the bit field:

BF bit pair: bit 1,0

00	Keep the previous value in the bio output register bit
01	Set the bio output register bit to zero
10	Set the bio output register bit to one
11	Complement the value in the bio output register bit

The transfer function is described below, where Qt is the previous value of an output register bit and Qt+1 is the new value:

Qt	Bit1	Bit0	Qt+1
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

When the bio register is read, the value of both the corresponding bio pin and the bio output register bit are read according to the following encoding of the bit field:

bit 0	Bio pin value
bit 1	Value of the bio output register bit

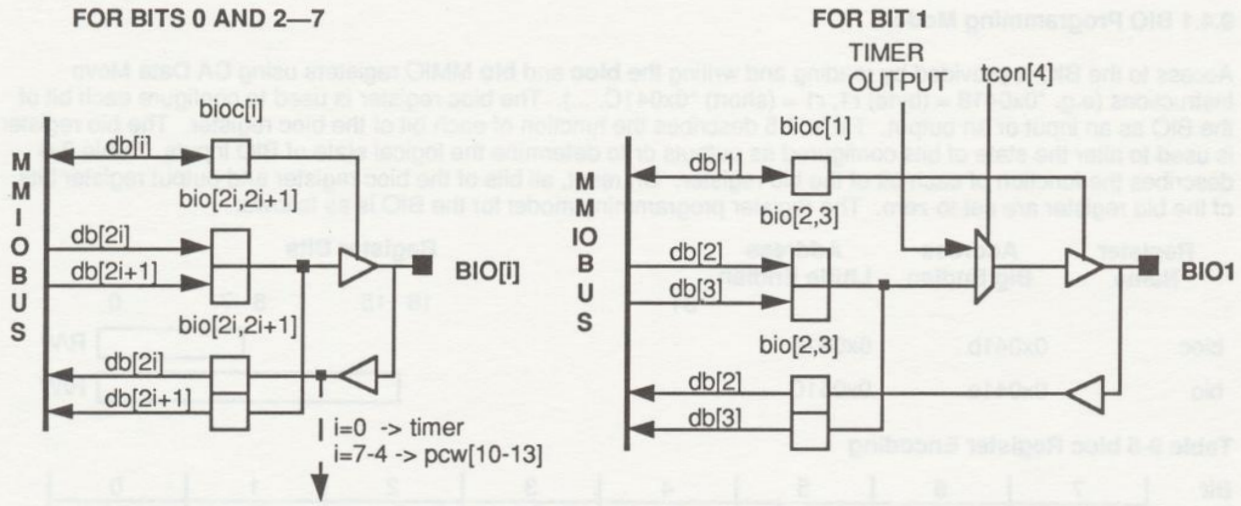


Figure 9-15 Bit I/O Bit Slice Diagrams

9.4.2 BIO Functional Description

Figure 9-15 shows bit slices for the two types of bits in the BIO. The structure for Bit 0 and bits 2—7 are the same, except that the BIO0 pin can be used as the timer source and BIO7—4 can be used to input the reset values of pcw[10—13]. Bit 1 has the additional capability of serving as the timer output, therefore an additional multiplexer controlled by tcon[4] is needed in this bit slice.

Each bit of the BIO port can be configured independently as either an input or an output by writing the desired value in the bioc (bit I/O configuration) register (see Table 9-5). If the pin is configured as an output, values written to the bio register are immediately reflected to the pin. The value of each output pin can be changed by writing the bio register with an appropriate pattern as described in Table 9-6. Each bio output register bit can be set to a 1 or a 0, toggled, or left unchanged. When the bio register is read, both the logic value of the pin and the value of the output register bit (the value that would be driven to the pin if the pin were configured as an output) are read.

9.4.3 BIO Timing Specifications

When configured as inputs by the bioc register, BIO pins are asynchronous inputs that are synchronized to the DSP3210 by an internal version of CKI. If an input makes a transition during the sample window, the level recognized by the processor is not predictable; however, the processor always resolves the latched level to either a logic one or zero before using it. Figure 9-16 shows the sampling window for BIO inputs. When configured as outputs, BIO pins change with respect to the falling edge of CKI. The timing relationship between the BIO outputs and CKI is also shown in Figure 9-16.

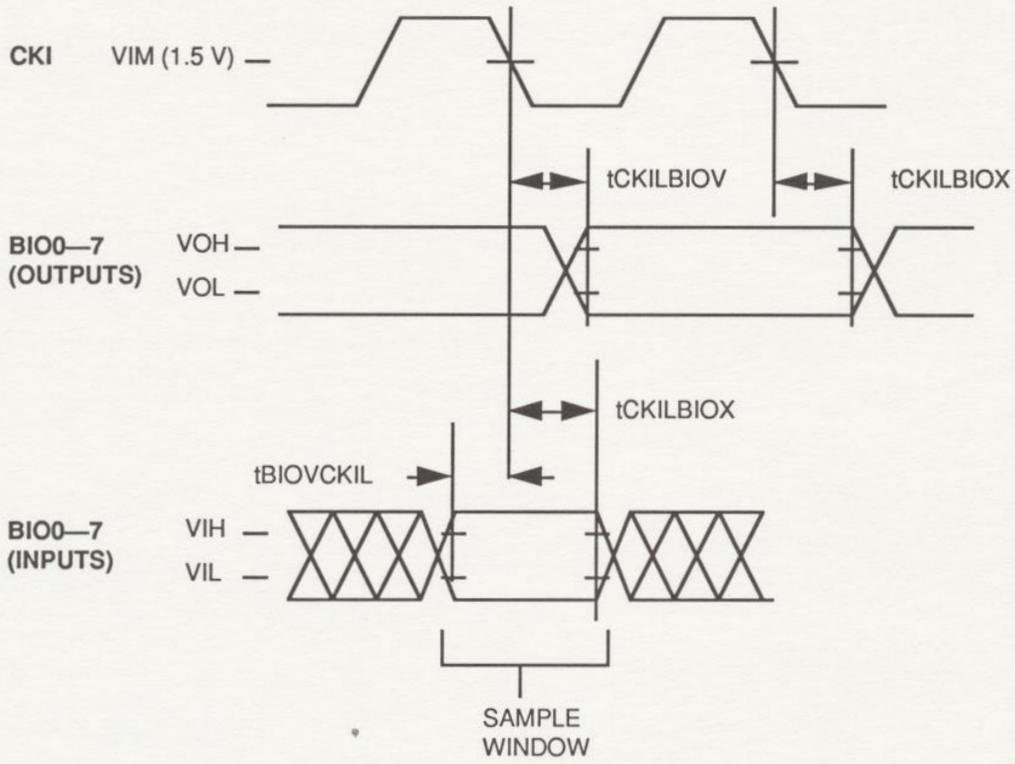


Figure 9-16 BIO Timing Specifications


Smile